

# A Bayesian Network Model for Interesting Itemsets

Jaroslav Fowkes\*

Charles Sutton\*

## Abstract

Mining itemsets that are the most interesting under a statistical model of the underlying data is a frequently used and well-studied technique for exploratory data analysis. The most recent models of interestingness are predominantly based on maximum entropy distributions over items or tile entries with itemset constraints, and while computationally tractable are not easily interpretable. We therefore propose the first, to the best of our knowledge, generative model over itemsets, in the form of a Bayesian network, and an associated novel measure of interestingness. Our model is able to efficiently infer interesting itemsets directly from the transaction database using structural EM, in which the E-step employs the greedy approximation to weighted set cover. Our approach is theoretically simple, straightforward to implement, trivially parallelizable and exhibits competitive performance as we demonstrate on both synthetic and real-world examples.

## 1 Introduction

Since its introduction over twenty years ago, itemset mining has been studied extensively, not just for its original application to market basket analysis but also for a variety of new applications ranging from mining data streams to mining bugs in source code [1].

The goal of itemset mining is to return a set of small itemsets that represent the most interesting patterns in a database of transactions. In most existing research, this is formalized by specifying an unsupervised data-driven *interestingness measure* for an itemset. The first such proposed measure was simply the frequency of an itemset in the database. Although appealing algorithmically, the list of frequent itemsets suffers from *pattern explosion*, i.e., is typically long, highly redundant, and difficult to understand [1].

In an attempt to address this problem, more sophisticated measures were devised. Rather than using simple *absolute measures* that score itemsets using only the data at hand, *statistical measures* that impose a statistical model on the data and measure how interesting itemsets are under the model, were developed. The vast majority of existing research uses two main classes of statistical model: independence models and binary models over dataset items/entries coupled with itemset constraints. While computationally tractable and algorithmically appealing, such models are not intuitive to interpret or easy to compose with other statistical models of the data. In particular, the popular class of maximum entropy models are prone to overfit the data and

so must be regularized with ad-hoc techniques such as MDL or BIC [1].

We therefore propose *Interesting Itemset Miner* (IIM), the first (to the best of our knowledge) *generative model*, a Bayesian network model that *directly infers* the itemsets that best explain the underlying data. The formulation of our generative model as a sequence of Bernoulli trials, one for each interesting itemset, ensures that the model has an implicit regularization built-in from the start and so does not require the use of any ad-hoc techniques such as MDL. Additionally, we also introduce a novel measure of itemset interestingness, defined as the ratio of transactions that an itemset explains under the model to those it supports.

The collection of interesting itemsets under IIM can be inferred efficiently using a structural EM framework [8]. One can think of our model as a probabilistic relative of some of the early work on itemset mining that formulates the task of finding interesting patterns as a covering problem [9, 20], except that in our work, the set cover problem is used to identify itemsets that cover a transaction *with maximum probability*. The set cover problem arises naturally within the E step of the EM algorithm. Our contributions in this paper are:

- IIM is the first *generative model* over itemsets, making it intuitively interpretable and easily composable with other models. Moreover, our model has implicit regularization bypassing the need for ad-hoc regularization techniques (Section 3.1).
- IIM is the first model to directly capture itemsets that ‘best explain’ a transaction database. That is to say, our model contains a latent indicator variable  $z_S$  that *directly infers* if itemset  $S$  was used to generate a transaction in the database (Section 3.2).
- We introduce a novel and intuitive model-derived measure of itemset interestingness, defined as the ratio of transactions that an itemset explains under the model to those it supports (Section 3.7).
- IIM is theoretically simple, straightforward to implement and trivially parallelizable. Our prototype implementation finds a hundred potentially interesting itemsets from a million transactions in under an hour on off-the-shelf hardware (Section 4.1).

On real-world datasets we find that the interesting itemsets seem to capture meaningful domain structure, e.g. representing phrases such as *anomaly detection* in a corpus of research papers, or regions such as *western US*

\*School of Informatics, University of Edinburgh, Edinburgh, EH8 9AB, UK, email: {jfowkes, csutton}@inf.ed.ac.uk

*states* in geographical data. Notably, we find that IIM returns a much more diverse list of itemsets than state of the art methods like KRIMP and CHARM (Table 1). Overall, our results suggest that the list of interesting itemsets found by our method is suitable for manual examination during exploratory data analysis.

Perhaps the primary contribution of this paper, however, is to show that — despite the enormous literature on itemset mining — a simple and efficient probabilistic model can identify more intuitive and useful patterns than standard approaches to this extremely well-studied problem.

## 2 Related Work

The first itemset mining algorithm was Apriori which recursively builds larger frequent itemsets from frequent singleton itemsets. While straightforward to implement, Apriori can spend a long time generating a large number of candidate frequent itemsets and scanning the database. The FPGrowth algorithm was devised to address these issues by mining frequent itemsets without candidate generation, by compressing the database into a data structure called an FP-tree (see Chapter 1 of [1] for a survey of frequent itemset algorithms).

A major issue with frequent itemset mining (FIM) is that of *pattern explosion*: a huge number of highly redundant frequent itemsets are retrieved if the given minimum support threshold is too low. In an attempt to address this, researchers devised *compact representations* of frequent itemsets, including maximal frequent, closed frequent, free and non-derivable itemsets, along with efficient algorithms to mine them such as CHARM [22]. While these approaches decrease the number of retrieved itemsets, they cannot solve the statistical pathologies of FIM (see Chapter 5 of [1]).

An orthogonal research direction has been to mine *tiles* instead of itemsets, i.e., subsets of rows *and columns* of the database viewed as binary transaction by item matrices. The natural extension of FIM is then to mine *large tiles*, i.e., sub-matrices with only 1s whose area is greater than a given minimum area threshold. The Tiling algorithm [9] is an example of an efficient implementation that uses the greedy algorithm for set cover. Note that there is a correspondence between tiles and itemsets: every large tile is a closed frequent itemset and thus such approaches suffer from the same issues as FIM. An extension is to mine *noisy tiles*, i.e., sub-matrices with predominantly 1s but some 0s [12].

In an attempt to tackle the inherent redundancy in FIM, modern approaches to itemset mining impose a statistical model on the data and measure how *interesting*, i.e., statistically significant, the itemsets are under the model. IIM also takes this approach. The sim-

plest model one can choose is the independence model, where all the individual items are assumed independent. Mined itemsets can then be ranked under this model using measures such as lift, Pearson’s  $\chi^2$  test, etc. (see [1], Ch. 5 for a survey).

A more sophisticated model better suited to itemsets is the *maximum entropy* (MaxEnt) model, a binary model over items which incorporates itemset frequencies as background knowledge and otherwise is as random as possible. The MaxEnt model is a natural extension of the independence model, which it reduces to when all dataset items are added as constraints. At the other extreme, adding all supported itemsets as constraints gives the empirical itemset distribution for the dataset. The MTV algorithm [14] uses the MaxEnt model to mine the set of top- $k$  itemsets with the highest likelihood under the model via an efficient convex bound which allows many candidate itemsets to be pruned and employs a method for more efficiently inferring the model itself. As the MaxEnt model enforces the constraint that an itemset must be used to explain a supported transaction, as opposed to *inferring* when this is the case as in our model, it tends to overfit the data and so MTV is forced to resort to model regularization techniques such as Minimum Description Length (MDL). Furthermore, due to the partitioning constraints necessary to keep computation feasible, MTV typically only finds in the order of tens of itemsets whereas our model has no such restriction.

It is also possible to use MDL directly: KRIMP [20] finds the subset of frequent itemsets that yields the best lossless compression of the database. While in principle this could be formulated as a set cover problem, the authors employ a fast heuristic that does not allow the itemsets to overlap, as opposed to IIM, even though one might expect that doing so could lead to better compression. In contrast, IIM employs a set cover framework to identify a set of itemsets that cover a transaction with highest probability. However, the main drawback of KRIMP is the need to mine a set of frequent itemsets in the first instance.

The MaxEnt model can also be extended to tiles, here known as the *Rasch* model, and unlike the itemset version inference takes polynomial time. However, the likelihood of the Rasch model monotonically decreases as more tiles are added, causing it to overfit and so model regularization techniques must be employed. Kontonasis and De Bie [12] use the Rasch model to find the most surprising set of noisy tiles by computing the likelihood of tile entries covered by the set. The inference problem then takes the form of weighted budgeted maximum set-cover, which can again be efficiently solved using the greedy algorithm.

In contrast to previous work, IIM maintains a generative model, in the form of a Bayesian network, directly over itemsets as opposed to indirectly over items. Existing Bayesian network models for itemset mining [11] have had limited success as modelling dependencies between all the items makes inference for larger datasets prohibitive. In IIM however, inference takes the form of a weighted set-cover problem, which can be solved efficiently using the greedy algorithm (Section 3.2).

The structure of IIM’s statistical model is similar to existing models in the literature such as Rephil ([17], §26.5.4) for topic modelling and QMR-DT [18] for medical diagnosis. Rephil is a multi-level graphical model with noisy-OR conditional probabilities used in Google’s AdSense system. As it is a proprietary model, very few details about it are available. But in general, topic models have different exploratory goals to itemset mining. Topic modelling aims at a set of patterns that globally organize the corpus, whereas itemsets represent more local patterns, e.g., correlations among small sets of items. QMR-DT is a bi-partite graphical model, with noisy-OR probabilities, used for inferring significant diseases based on medical findings. By contrast, the main contribution of our paper is to show that a binary latent variable model can be useful for selecting itemsets for exploratory data analysis.

### 3 Interesting Itemset Mining

In this section we will formulate the problem of identifying a set of interesting itemsets that are useful for explaining a database (i.e., sequence) of transactions. First we will define some preliminary concepts and notation. An *item*  $i$  is an element of the universe  $U = \{1, 2, \dots, n\}$  that indexes database attributes. A *transaction*  $X$  is a subset of the universe  $U$  and an *itemset*  $S$  is simply a set of items  $i$ . The set of interesting itemsets  $\mathcal{I}$  we wish to determine is therefore a subset of the power set (set of all possible subsets) of the universe. Further, we say that an itemset  $S$  *supports* a transaction  $X$  if  $S \subset X$ .

**3.1 Bayesian Network Model** We propose a simple directed graphical model for generating a database of transactions  $X^{(1)}, \dots, X^{(m)}$  from a set  $\mathcal{I}$  of interesting itemsets. The parameters of our model are Bernoulli probabilities  $\pi_S$  for each interesting itemset  $S \in \mathcal{I}$ . The generative story for our model is, independently for each transaction  $X$  in the database:

1. For each itemset  $S \in \mathcal{I}$ , decide independently whether to include  $S$  in the transaction, i.e., sample

$$z_S \sim \text{Bernoulli}(\pi_S).$$

2. Set the transaction to be the set of items in all the

itemsets selected above, i.e.,

$$X = \bigcup_{S|z_S=1} S.$$

Note that the model allows individual items to be generated multiple times from different itemsets, e.g. *eggs* could be generated both as part of a breakfast itemset  $\{\text{bacon}, \text{eggs}\}$  and as as part of a cake itemset  $\{\text{flour}, \text{sugar}, \text{eggs}\}$ .

Now given a set of itemsets  $\mathcal{I}$ , let  $\mathbf{z}, \boldsymbol{\pi}$  denote the vectors of  $z_S, \pi_S$  for all  $S \in \mathcal{I}$ . Assuming  $\mathbf{z}, \boldsymbol{\pi}$  are fully determined, it is evident from the generative model that the probability of generating a transaction  $X$  is

$$p(X, \mathbf{z} | \boldsymbol{\pi}) = \begin{cases} \prod_{S \in \mathcal{I}} \pi_S^{z_S} (1 - \pi_S)^{1-z_S} & \text{if } X = \bigcup_{z_S=1} S, \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

since each  $z_S \sim \text{Bernoulli}(\pi_S)$ .

**3.2 Inference** Assuming the latent variables  $\boldsymbol{\pi}$  in the model are known, we can infer  $\mathbf{z}$  for a specific transaction  $X$  by maximizing the posterior distribution  $p(\mathbf{z} | X, \boldsymbol{\pi})$  over  $\mathbf{z}$ :

$$\begin{aligned} \max_{\mathbf{z}} \quad & \prod_{S \in \mathcal{I}} \pi_S^{z_S} (1 - \pi_S)^{1-z_S} \\ \text{s.t. } \quad & X = \bigcup_{S|z_S=1} S. \end{aligned} \quad (3.2)$$

Taking logs and rewriting (3.2) in a more standard form we obtain

$$\begin{aligned} \min_{\mathbf{z}} \quad & \sum_{S \in \mathcal{I}} z_S (-\ln(\pi_S)) + (1 - z_S) (-\ln(1 - \pi_S)) \\ \text{s.t. } \quad & \sum_{S|i \in S} z_S \geq 1 \quad \forall i \in X \\ & z_S \in \{0, 1\} \quad \forall S \in \mathcal{I} \end{aligned} \quad (3.3)$$

which is (up to a penalty term) the weighted set-cover problem (see e.g. [13], §16.1) with weights  $w_S \in \mathbb{R}^+$  given by  $w_S := -\ln(\pi_S)$ . This is an NP-hard problem in general and so impractical to solve directly in practice. It is important to note that the weighted set cover problem is a special case of minimizing a linear function subject to a submodular constraint<sup>1</sup>, which we formulate as follows (cf. [21]). Given the set of interesting itemsets  $\mathcal{T} := \{S \in \mathcal{I} | S \subset X\}$  that support the transaction, a real-valued weight  $w_S$  for each itemset  $S \in \mathcal{T}$  and a non-decreasing submodular function  $f : 2^{\mathcal{T}} \rightarrow \mathbb{R}$ , the aim is to find a covering  $C \subset \mathcal{T}$

<sup>1</sup>Note that the posterior  $p(\mathbf{z} | X)$  would not be submodular if we were to use a noisy-OR model for the conditional probabilities.

of minimum total weight, i.e., such that  $f(\mathcal{C}) = f(\mathcal{T})$  and  $\sum_{S \in \mathcal{C}} w_S$  is minimized. For weighted set cover we simply define  $f(\mathcal{C})$  to be the number of items in  $\mathcal{C}$ , i.e.,  $f(\mathcal{C}) := |\cup_{S \in \mathcal{C}} S|$ . Note that  $f(\mathcal{T}) = |X|$  by construction.

We can then approximately solve the weighted set cover problem (3.3) using the greedy approximation algorithm for submodular functions. The greedy algorithm builds a covering  $\mathcal{C}$  by repeatedly choosing an itemset  $S$  that minimizes the weight  $w_S$  divided by the number of items in  $S$  not yet covered by the covering. In order to minimize CPU time spent solving the weighted set cover problem, we cache the itemsets and coverings for each transaction as needed.

It has been shown [3] that the greedy algorithm achieves a  $\ln|X| + 1$  approximation ratio to the weighted set cover problem and moreover the following inapproximability theorem shows that this is essentially the best possible approximation ratio.

**THEOREM 3.1.** ([6]) *There is no  $(1 - o(1))\ln|X|$ -approximation algorithm to the weighted set cover problem unless  $\text{NP} \subseteq \text{DTIME}(|X|^{O(\log \log |X|)})$ , i.e., unless NP has slightly superpolynomial time algorithms.*

The runtime complexity of the greedy algorithm is  $O(|X||\mathcal{T}|)$ , however by maintaining a priority queue this can be improved to  $O(|X|\log|\mathcal{T}|)$  (see e.g. [4]). Note that there is also an  $O(|X||\mathcal{T}|)$ -runtime primal-dual approximation algorithm [2], however this has an approximation order of  $f = \max_i |\{S \mid i \in S\}|$ , i.e., the frequency of the most frequent element, which would inevitably be worse in our case.

**3.3 Learning** Given a set of itemsets  $\mathcal{I}$ , consider now the case where both variables  $\mathbf{z}, \boldsymbol{\pi}$  in the model are unknown. In this case we can use the hard EM algorithm [5] for parameter estimation with latent variables. The hard EM algorithm in our case is merely a simple layer on top of the inference algorithm (3.3). Suppose there are  $m$  transactions  $X^{(1)}, \dots, X^{(m)}$  with supporting sets of itemsets  $\mathcal{T}^{(1)}, \dots, \mathcal{T}^{(m)}$ , then the hard EM algorithm is given in Algorithm 1. To initialize  $\boldsymbol{\pi}$ , a natural choice is simply the support (relative frequency) of each itemset in  $\mathcal{I}$ .

**3.4 Inferring new itemsets** We infer new itemsets using structural EM [8], i.e., we add a candidate itemset  $S'$  to  $\mathcal{I}$  if doing so improves the optimal value  $\bar{p}$  of the problem (3.3) averaged across transactions. Interestingly, there is an implicit regularization effect here. Observe from (3.3) that when a new candidate  $S'$  is added to the model, a corresponding term  $\ln(1 - \pi_{S'})$  is added to the log-likelihood of all transactions that  $S'$  does not support. For large databases, this amounts to

---

#### Algorithm 1 HARD-EM

---

**Input:** Set of itemsets  $\mathcal{I}$  and initial estimates  $\boldsymbol{\pi}^{(0)}$   
 $k \leftarrow 0$   
**do**  
 $k \leftarrow k + 1$   
E-STEP:  $\forall X^{(j)}$  solve (3.3) to get  $z_S^{(j)} \forall S \in \mathcal{T}_j$   
M-STEP:  $\pi_S^{(k)} \leftarrow \frac{1}{m} \sum_{j=1}^m z_S^{(j)} \forall S \in \mathcal{I}$   
**while**  $\|\boldsymbol{\pi}^{(k-1)} - \boldsymbol{\pi}^{(k)}\| > \varepsilon$   
Remove from  $\mathcal{I}$  sets  $S$  with  $\pi_S = 0$   
**return**  $\mathcal{I}, \boldsymbol{\pi}^{(k)}$

---

a significant penalty on candidates in practice.

To get an estimate of maximum benefit to including candidate  $S'$ , we must carefully choose an initial value of  $\pi_{S'}$  that is not too low, to avoid getting stuck in a local optimum. To infer a good  $\pi_{S'}$ , we force the candidate  $S'$  to explain all transactions it supports by initializing  $\pi_{S'} = 1$  and update  $\pi_{S'}$  with the probability corresponding to its actual usage once we have inferred all the coverings. Given a set of itemsets  $\mathcal{I}$  and corresponding probabilities  $\boldsymbol{\pi}$  along with transactions  $X^{(1)}, \dots, X^{(m)}$ , each iteration of the structural EM algorithm is as follows.

---

#### Algorithm 2 STRUCTURAL-EM (one iteration)

---

**Input:** Itemsets  $\mathcal{I}$ ,  $\boldsymbol{\pi}$ , optima  $p^{(j)}$  of (3.3)  $\forall X^{(j)}$   
Set profit  $\bar{p} \leftarrow \frac{1}{m} \sum_{j=1}^m p^{(j)}$   
**do**  
Generate candidate  $S'$  using CANDIDATE-GEN  
 $\mathcal{I} \leftarrow \mathcal{I} \cup \{S'\}, \pi_{S'} \leftarrow 1$   
E-STEP:  $\forall X^{(j)}$  solve (3.3) to get  $z_S^{(j)} \forall S \in \mathcal{T}_j$   
M-STEP:  $\pi'_S \leftarrow \frac{1}{m} \sum_{j=1}^m z_S^{(j)} \forall S \in \mathcal{I}$   
 $\forall X^{(j)}$ , solve (3.3) using  $\pi'_S, z_S^{(j)} \forall S \in \mathcal{T}_j$   
to get the optimum  $p^{(j)}$   
Set new profit  $\bar{p}' \leftarrow \frac{1}{m} \sum_{j=1}^m p^{(j)}$   
 $\mathcal{I} \leftarrow \mathcal{I} \setminus \{S'\}$   
**while**  $\bar{p}' \leq \bar{p}$  {until one good candidate found}  
 $\mathcal{I} \leftarrow \mathcal{I} \cup \{S'\}$   
**return**  $\mathcal{I}, \boldsymbol{\pi}'$

---

In practice we store the set of candidates that have been rejected by STRUCTURAL-EM and check each potential candidate against this set for efficiency.

**3.5 Candidate generation** The STRUCTURAL-EM algorithm (Algorithm 2) requires a method to generate new candidate itemsets  $S'$  that are to be considered for inclusion in the set of interesting itemsets  $\mathcal{I}$ . One possibility would be to use the Apriori algorithm to recursively suggest larger itemsets starting from singletons,

however preliminary experiments found this was not the most efficient method. For this reason we take a slightly different approach and recursively combine the interesting itemsets in  $\mathcal{I}$  with the *highest support first* (Algorithm 3). In this way our candidate generation algorithm is more likely to propose viable candidate itemsets earlier and in practice we find that this heuristic works well. We did try the pruning potential itemset pairs to join using a  $\chi^2$ -test, however this substantially slowed down the algorithm and barely improved the model likelihood.

---

**Algorithm 3** CANDIDATE-GEN

---

**Input:** Itemsets  $\mathcal{I}$ , cached supports  $\sigma$ , queue length  $q$   
**if**  $\nexists$  priority queue  $\mathcal{Q}$  for  $\mathcal{I}$  **then**  
    Initialize  $\sigma$ -ordered priority queue  $\mathcal{Q}$   
    Sort  $\mathcal{I}$  by decreasing itemset support using  $\sigma$   
    **for** all distinct pairs  $S_1, S_2 \in \mathcal{I}$ , highest ranked first **do**  
        Generate candidate  $S' = S_1 \cup S_2$   
        Cache support of  $S'$  in  $\sigma$  and add  $S'$  to  $\mathcal{Q}$   
        **if**  $|\mathcal{Q}| = q$  **break**  
    **end for**  
**end if**  
    Pull highest-ranked candidate  $S'$  from  $\mathcal{Q}$   
**return**  $S'$

---

In order to determine the supports of the itemsets to be combined, we store the transaction database in a Memory-Efficient Itemset Tree (MEI-TREE) [7] and query the tree for the support of a given itemset. A MEI-TREE stores itemsets in a tree structure according to their prefixes in a memory efficient manner. To minimize the memory usage of the MEI-TREE further, we first sort the items in order of decreasing support (as in the FPGrowth algorithm) as this often results in a sparser tree [10]. Note that a MEI-TREE is essentially an FP-tree [10] with node-compression and without node-links for nodes containing the same item. An itemset support query on the MEI-TREE efficiently searches the tree for all occurrences of the given itemset and adds up their supports (see Figure 4 in [7] for the actual algorithm). With the wide availability of 100GB+ shared memory systems, it is reasonable to expect the MEI-TREE to fit into memory for all but the largest of datasets. The queue length parameter in the CANDIDATE-GEN algorithm effectively imposes a limit on the number of iterations the algorithm can spend suggesting candidate itemsets.

**3.6 Mining Interesting Itemsets** Our complete interesting itemset mining (IIM) algorithm is given in Algorithm 4. Note that the HARD-EM parameter optimization step need not be performed at every iteration,

---

**Algorithm 4** IIM (Interesting Itemset Miner)

---

**Input:** Database of transactions  $X^{(1)}, \dots, X^{(m)}$   
    Initialize  $\mathcal{I}$  with singletons,  $\pi$  with their supports  
    Build MEI-TREE from transaction database  
    **while** not converged **do**  
        Add itemsets to  $\mathcal{I}$ ,  $\pi$  using STRUCTURAL-EM  
        Optimize parameters for  $\mathcal{I}$ ,  $\pi$  using HARD-EM  
    **end while**  
**return**  $\mathcal{I}, \pi$

---

in fact it is more efficient to suggest several candidate itemsets before optimizing the parameters. As all operations on transactions in our algorithm are trivially parallelizable, we perform the  $E$  and  $M$ -steps in both the hard and structural EM algorithms in parallel.

**3.7 Interestingness Measure** Now that we have inferred the model variables  $\mathbf{z}, \pi$ , we are able to use them to rank the retrieved itemsets in  $\mathcal{I}$ . There are two natural rankings one can employ, and both have their strengths and weaknesses. The obvious approach is to rank each itemset  $S \in \mathcal{I}$  according to its probability under the model  $\pi_S$ , however this has the disadvantage of strongly favouring frequent itemsets over rare ones, an issue we would like to avoid. Instead, we prefer to rank the retrieved itemsets according to their *interestingness* under the model, that is the ratio of transactions they explain to transactions they support. One can think of interestingness as a measure of how necessary the itemset is to the model: the higher the interestingness, the more supported transactions the itemset explains. Thus interestingness provides a more balanced measure than probability, at the expense of missing some frequent itemsets that only explain some of the transactions they support. We define interestingness formally as follows.

**DEFINITION 3.1.** *The interestingness of an itemset  $S \in \mathcal{I}$  retrieved by IIM (Algorithm 4) is defined as*

$$int(S) = \frac{\sum_{j=1}^m z_S^{(j)}}{supp(S)}$$

*and ranges from 0 to 1 (least to most interesting).*

Any ties in the ranking can be broken using the itemset probability  $\pi_S$ .

**3.8 Similarity to existing models** The astute reader may have noted that IIM's underlying statistical model is similar to existing models for itemsets proposed in the literature (cf. Section 2), however there are crucial differences which we will highlight in this section. For a transaction  $X$ , the MaxEnt distribution over itemsets

$S \in \mathcal{I}$ , as used by MTV, can be written (cf. [14]):

$$p(X) = \pi_0 \prod_{S \in \mathcal{I}} \pi_S^{\mathbf{1}_X(S)}$$

where the indicator function  $\mathbf{1}_X(S) = 1$  if  $X$  supports  $S$  and 0 otherwise. There are two key differences here compared to IIM (3.1): a) if an itemset is present in the MaxEnt model *it must be used* to explain a supported transaction, contrast this with IIM where there is a latent variable  $z_S^{(j)}$  for each transaction  $X^{(j)}$  that *infers if an itemset is used* to explain the transaction; and b) IIM’s use of a Bernoulli distribution for each itemset leads to an *implicit regularization* term  $(1 - \pi_S)^{1 - z_S^{(j)}}$  in (3.1) meaning that unlike MTV we do not need to employ ad-hoc model regularization techniques.

KRIMP by contrast, uses an itemset independence model, which for an itemset  $S \in \mathcal{I}$  is given by (cf. [20]):

$$p(S) = \sum_{j=1}^m z_S^{(j)} / \sum_{I \in \mathcal{I}} \sum_{k=1}^m z_I^{(k)}$$

where the  $z_S^{(j)}$ , and therefore itemset coverings for  $X^{(j)}$ , are determined using a *heuristic approximation*. That is, unlike IIM, the itemset coverings are not chosen to maximise the probability under the statistical model. Instead, for each transaction  $X$ , supported itemsets  $S \in \mathcal{I}$  are chosen according to *decreasing size and support* until all elements of  $X$  are covered. Additionally, itemsets in the covering are not allowed to overlap, in contrast to IIM which does allow overlap if necessary (although our model discourages it). Moreover,  $\mathcal{I}$  is initialised with singletons and candidate *frequent itemsets from a pre-computed set* are added to  $\mathcal{I}$  one at a time and retained only if the surprisal under the above model improves. In contrast, IIM uses structural EM and is able to *add candidate itemsets on the fly*.

One can also think of IIM as a probabilistic tiling method: each interesting itemset  $S \in \mathcal{I}$  can be thought of as a binary submatrix of transactions for which  $z_S = 1$  by items in  $S$ , where the choice of items and transactions in the tile are *inferred directly* from IIM’s statistical model. That is, IIM formulates the inference problem (3.3) as *weighted set cover* for *each transaction* where the weights correspond to *itemset probabilities*. This is in contrast to existing tiling methods: [9] finds  $k$  tiles covering the largest number of database entries and is thus an instance of *maximum coverage*. [12], extends this to inferring a covering of noisy tiles using *budgeted maximum coverage*, that is, finding a covering that maximizes the sum of the *surprisal* of each tile, under a MaxEnt model constrained by expected row and column margins, subject to the sum of the *description lengths* of each tile being smaller than a given budget.

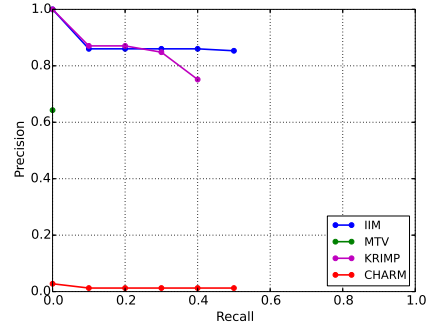


Figure 1: Precision against recall for IIM, MTV, KRIMP and CHARM on our synthetic database, using the top- $k$  itemsets as a threshold<sup>2</sup> (note that MTV is a single point at the far left).

## 4 Numerical Experiments

In this section we perform a comprehensive qualitative and quantitative evaluation of IIM. On synthetic datasets we show that IIM returns a list of itemsets that does not contain many spurious correlations, is largely non-redundant and includes rare itemsets. Moreover, we show that IIM scales linearly with the number of transactions. On a set of real-world datasets (see Table 2) we show that IIM finds itemsets that are consistent, interpretable and highly relevant to the problem at hand.

### 4.1 Quantitative Evaluation

**Spurious Itemsets** The set-cover formulation of the IIM algorithm (3.3) naturally favours adding itemsets to the model whose items co-occur in the transaction database. One would therefore expect IIM to largely avoid creating itemsets of uncorrelated items and so generate more meaningful itemsets. To verify this is the case, we ran IIM, MTV [14], KRIMP [20] and CHARM with  $\chi^2$ -test ranking [22] on a synthetic transaction database. To obtain a realistic synthetic database, we first trained IIM on the plants dataset [19], yielding a model with 328 interesting itemsets. Then we created a synthetic database by sampling 10,000 transactions from this model. Figure 1 shows the precision-recall curve for each algorithm using the top- $k$  itemsets as a threshold ( $k$  ranges over all the itemsets returned). One can clearly see that IIM and KRIMP perform best and this strongly implies they largely avoid the issue of retrieving items that co-occur independently in the database, but are both frequent, as a single itemset. MTV returns a very small pattern set and consequently has very small recall, however it does have high precision suggesting the returned itemsets are mostly not redundant. In contrast, the set of closed itemsets returned by CHARM is not only large but also hugely redundant. We used a minimum support of 0.0575 for MTV, KRIMP and CHARM as used in [14] for the plants dataset.

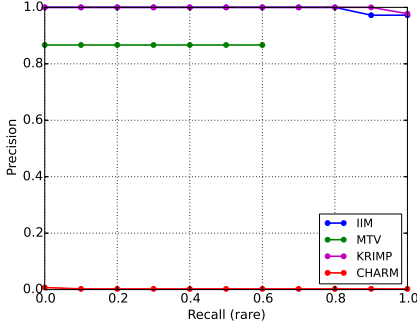


Figure 2: Precision against *rare itemset* recall for IIM, MTV, KRIMP and CHARM on our synthetic database, using the top- $k$  itemsets as a threshold<sup>2</sup>.

**Rare Itemsets** We also evaluated the ability of IIM, MTV, KRIMP and CHARM to find rare itemsets, i.e., itemsets with low support, in a background distribution. We took the interesting itemsets mined from the plants database in the previous section, added in 30 itemsets with a relative support of 0.05 (each containing two items not already present in the dataset) and generated a 10,000 transaction synthetic database using our probabilistic model. We then ran the aforementioned algorithms, using a minimum support of 0.04 for all but IIM (which does not require a support threshold). We plot the background and rare itemset precision against rare itemset recall in Figure 2, using the top- $k$  itemsets as a threshold. One can see that both IIM and KRIMP perform better than MTV and are able to recover all the rare itemsets while maintaining very high precision. While CHARM is able to recover all the rare itemsets, it has very low recall since the list of closed frequent itemsets (ranked by  $\chi^2$ ) it returns is huge and highly redundant, as it has no way of pruning variants of itemsets that identify single statistically significant concepts.

**Redundancy** Now we evaluate on real-world data whether IIM returns a less redundant list of itemsets than the other state-of-the-art methods. A suitable measure of redundancy for a single itemset is the minimum symmetric difference between it and the other itemsets in the list. Averaging this across all itemsets in the list, we obtain the *average inter-itemset distance*. We therefore ran IIM, KRIMP and CHARM on the

	Plants	Mammals	ICDM	Uganda
IIM	3.50	5.30	3.66	3.72
KRIMP	1.53	2.02	2.22	2.24
CHARM	1.53	1.52	1.47	1.45

Table 1: Average inter-itemset distance of the top 100 non-singleton itemsets for the given methods on the datasets from Table 2 (MTV is excluded as it could not return 100 itemsets).

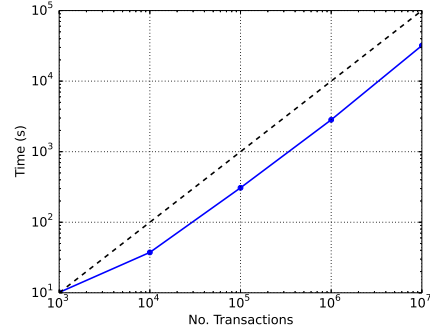


Figure 3: IIM scaling as the number of transactions in our synthetic database increases.

datasets in Table 2. This enabled us to calculate, for each dataset, the average inter-itemset distance of the top 100 non-singleton itemsets, which we report in Table 1. For CHARM, we took the top 100 non-singleton itemsets ranked according to  $\chi^2$  from the top 100,000 frequent itemsets it returned (as the  $\chi^2$  calculation would be prohibitively slow otherwise). One can clearly see that the top IIM itemsets have a larger inter-itemset distance on average, and are therefore less redundant, than the KRIMP or CHARM itemsets. Moreover, the top CHARM  $\chi^2$ -ranked itemsets are more redundant than KRIMP as expected. Note that we excluded MTV as it was not able to return 100 non-singleton itemsets for any of the datasets.

**Scalability** Finally, we investigated the scaling of IIM as the number of transactions in the database increases. We used the model trained on the plants dataset from the previous sections to generate synthetic transaction databases of various sizes. We then ran IIM for 100 iterations on these databases (Figure 3) and one can see the transaction scaling is linear as expected. Our prototype implementation can process one million transactions in 30 seconds on 64 cores each iteration, so there is reason to hope that a more highly tuned implementation could scale to even larger datasets.

**4.2 Qualitative Evaluation** To evaluate the quality of the itemsets produced by our method, we ran IIM for 1,000 iterations with a priority queue size of 100,000 on several real-world datasets (see Table 2 for a summary). We also compared to MTV and KRIMP but not CHARM as the top itemsets it returned were extremely redundant (as expected from the previous section). Note that we chose not to compare to the tiling methods from [9, 12] as they have been shown to underperform on the ICDM dataset [14].

**Plants and Mammals Datasets** See [19, 16] resp.

<sup>2</sup>Each plotted curve is the 11-point interpolated precision i.e., the interpolated precision at 11 equally spaced recall points between 0 and 1 (inclusive), see [15], §8.4 for details.



For both datasets, IIM, MTV and KRIMP find itemsets that are spatially coherent, but as we showed in Table 1, those returned by IIM are far less redundant. In particular, our interestingness measure enables IIM to rank correlated itemsets above singletons and rare itemsets above frequent ones, in contrast to MTV or KRIMP. For example, for the plants dataset, the top itemset retrieved by IIM is  $\{\textit{Puerto Rico}, \textit{Virgin Islands}\}$  whereas MTV returns  $\{\textit{Puerto Rico}\}$ , not associating it with the *Virgin Islands* (which are adjacent) until the 20th ranked itemset. For the mammals dataset the top two non-singleton IIM itemsets are a group of four mammals that coexist in Scotland and Ireland and a group of ten mammals that coexist on Sweden’s border with Norway. By contrast, the top four KRIMP itemsets are lists of some of the most common and well-known mammals in Europe (see the supplementary material for details).

**ICDM Dataset** In this dataset [12], each transaction is the abstract for an ICDM paper and each item is a stemmed word in the abstract, excluding stop-words. We show the top ten non-singleton itemsets returned by IIM (ranked according to interestingness), MTV (ranked according to probability) and KRIMP (ranked according to usage) in Table 3. One can see that the IIM itemsets are more informative since they contain interesting collocations not found by the other algorithms e.g.  $\{\textit{privaci}, \textit{sensit}\}$ ,  $\{\textit{main}, \textit{memori}\}$ . The IIM itemsets also suggest that the stemmer used to process the dataset could be improved, as we retrieve the itemsets  $\{\textit{parameter}, \textit{parameters}\}$ ,  $\{\textit{sequenc}, \textit{sequential}\}$ ,  $\{\textit{ensemble}, \textit{ensembles}\}$  and  $\{\textit{subset}, \textit{subsets}\}$ .

**Uganda Dataset** This dataset consists of Facebook messages taken from a set of public Uganda-based pages with substantial topical discussion (pages of newspapers, radio stations, government ministries and embassies) over a period of three months. Each transaction in the dataset is an English language message and each item is a stemmed English word from the message. The top six non-singleton itemsets found by the algorithms are shown in Table 4; the IIM itemsets provide much more information about the topics of the messages than those from MTV and KRIMP. Figure 4 plots the mentions of each of the top IIM itemsets per day. As one can see, usage of the top itemsets display

Dataset	Items	Transactions	Itemsets†	Runtime
Plants	70	34,781	259	27 min
Mammals	194	2,670	359	22 min
ICDM	4,976	859	798	163 min
Uganda	33,278	124,566	928	1086 min

Table 2: Summary of the real datasets used and IIM results after 1,000 iterations. †excluding singleton itemsets.

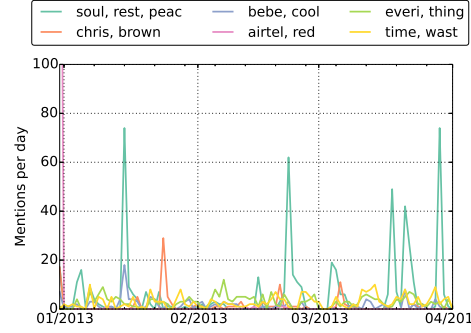


Figure 4: Mentions per day of the top six non-singleton IIM itemsets from the Uganda messages dataset over three months.

IIM	MTV	KRIMP
soul, rest, peace	heal, jesus, amen	whi, ?
chris, brown	god, amen	?, !
bebe, cool	2, 4	2, 4
airtel, red	whi, ?	wat, ?
everi, thing	god, heal	time, !
time, wast	2, !	soul, rest, peace

Table 4: Top six non-singleton Uganda itemsets for each method.

temporal structure (and exhibit spikes of popularity), even though our model does not explicitly capture this. Of particular interest are the large spikes of the itemset  $\{\textit{soul}, \textit{rest}, \textit{peac}\}$  corresponding to notable deaths: wealthy businessman James Mulwana on the 15th January, President Museveni’s father on the 22nd February and six school students in a traffic accident on the 29th March. Also of interest are the 285 mentions of  $\{\textit{airtel}, \textit{red}\}$  on New Year’s Eve corresponding to mobile provider Airtel’s Red Christmas competition for 10K worth of airtime. The spike of  $\{\textit{bebe}, \textit{cool}\}$  on the 15th January corresponds to the Ugandan musician’s wedding announcement and the spike on the 24th January of  $\{\textit{chris}, \textit{brown}\}$  refers to many enthusiastic mentions of the popular American singer that day. The last two itemsets capture that *everything* and *time-wasting* are common phrases.

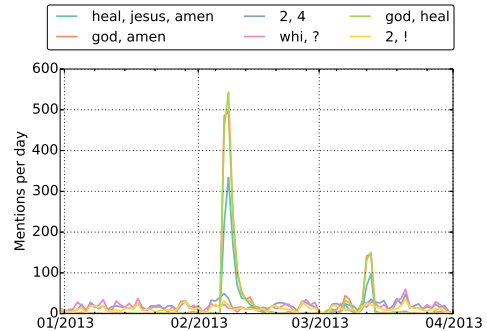


Figure 5: Mentions per day of the top six non-singleton MTV itemsets from the Uganda messages dataset over three months.



IIM	MTV	KRIMP	IIM not MTV/KRIMP
associ rule	synthetic real	algorithm experiment result set	sequenc sequential
local global	associ rule mine	decis tree	linear discriminant analysi
support vector machin svm	frequent pattern mine algorithm	high dimensional	synthetic real life
parameter parameters	frequent itemset mine	inform model	ensemble ensembles
anomaly detect	support vector machin	associ rule mine	event occur
sequenc sequential	naiv bayes	synthetic real	frequent item itemset transact
linear discriminant analysi	state art	time seri	learner train
synthetic real life	nearest neighbor	feature select	main memori
background knowledg	play role	knowledge discoveri	privaci sensit
semi supervised	linear discriminant analysi lda	method find	subset subsets

Table 3: Top ten non-singleton ICDM itemsets as found by IIM, MTV and KRIMP as well as those found by IIM *but not* MTV/KRIMP. Note that *MTV not IIM/KRIMP* and *KRIMP not IIM/MTV* were both empty, lending further support to IIM’s statistical model.

In comparison, the top-six MTV itemsets are plotted in Figure 5. One can see that the itemsets  $\{heal, \text{jesus}, \text{amen}\}; \{god, \text{amen}\}$  and  $\{god, heal\}$  substantially overlap and are strongly correlated with each other, sharing a large spike on the 8th February and a smaller spike on the 11th March. The remaining itemsets exhibit no interesting spikes as one would expect. The top six KRIMP itemsets in Table 4 all displayed random time evolution (as one would expect) except for  $\{soul, rest, \text{peac}\}$  which we have already encountered.

## 5 Conclusions

We presented a generative model that directly infers itemsets that best explain a transaction database along with a novel model-derived measure of interestingness and demonstrated the efficacy of our approach on both synthetic and real-world databases. In future we would like to extend our approach to directly inferring the association rules implied by the itemsets and parallelize our approach to large clusters so that we can efficiently scale to much larger databases.

## References

- [1] C. Aggarwal and J. Han. *Frequent Pattern Mining*. Springer, 2014.
- [2] R. Bar-Yehuda and S. Even. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981.
- [3] V. Chvátal. A greedy heuristic for the set-covering problem. *Math. O.R.*, 4(3):233–235, 1979.
- [4] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.
- [5] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, pages 1–38, 1977.
- [6] U. Feige. A threshold of  $\ln n$  for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.
- [7] P. Fournier-Viger, E. Mwamikazi, T. Gueniche, and U. Faghihi. MEIT: Memory Efficient Itemset Tree for targeted association rule mining. In *Advanced Data Mining and Applications*, volume 8347 of *Lecture Notes in Computer Science*, pages 95–106. Springer, 2013.
- [8] N. Friedman. The Bayesian structural EM algorithm. In *UAI*, pages 129–138, 1998.
- [9] F. Geerts, B. Goethals, and T. Mielikäinen. Tiling databases. In *Discovery science*, pages 278–289. Springer, 2004.
- [10] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD Record*, volume 29, pages 1–12. ACM, 2000.
- [11] S. Jaroszewicz and D. A. Simovici. Interestingness of frequent itemsets using Bayesian networks as background knowledge. In *SIGKDD*, pages 178–186. ACM, 2004.
- [12] K.-N. Kontonassios and T. De Bie. An information-theoretic approach to finding informative noisy tiles in binary databases. In *SDM*, pages 153–164. SIAM, 2010.
- [13] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Algorithms and Combinatorics. Springer, 2012.
- [14] M. Mampaey, J. Vreeken, and N. Tatti. Summarizing data succinctly with the most informative itemsets. *TKDD*, 6(4):16, 2012.
- [15] C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [16] A. Mitchell-Jones, G. Amori, W. Bogdanowicz, B. Kryštufek, P. Reijnders, F. Spitzenberger, M. Stubbe, J. Thissen, V. Vohralík, and J. Zima. *The Atlas of European Mammals*. T & AD Poyser, 1999.
- [17] K. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.
- [18] M. A. Shwe, B. Middleton, D. Heckerman, M. Henrion, E. Horvitz, H. Lehmann, and G. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. *Methods of information in Medicine*, 30(4):241–255, 1991.
- [19] USDA. The PLANTS Database, 2008.
- [20] J. Vreeken, M. Van Leeuwen, and A. Siebes. KRIMP: mining itemsets that compress. *Data Mining and Knowledge Discovery*, 23(1):169–214, 2011.
- [21] N. Young. Greedy set-cover algorithms (1974–1979, Chvátal, Johnson, Lovász, Stein). In M. Kao, editor, *Encyclopedia of Algorithms*, pages 379–381. Springer, 2008.
- [22] M. J. Zaki and C.-J. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In *SDM*, volume 2, pages 457–473. SIAM, 2002.